SYSTEM, METHOD, AND COMPUTER PROGRAM PRODUCT FOR REPRESENTING OBJECT RELATIONSHIPS IN A MULTIDIMENSIONAL SPACE

Inventors: Dimitris K. Agrafiotis
Dmitrii N. Rassokhin
Victor S. Lobanov
F. Raymond Salemme

CROSS-REFERENCE TO RELATED APPLICATIONS

This application claims the benefit of U.S. Provisional Application No. 60/191,108, filed March 22, 2000 (incorporated in its entirety herein by reference).

BACKGROUND OF THE INVENTION

Field of the Invention

[0001] The invention described herein relates to information representation, information cartography and data mining. The present invention also relates to pattern analysis and representation, and, in particular, representation of object relationships in a multidimensional space.

Related Art

[0002] Reducing the dimensionality of large multidimensional data sets is an important objective in many data mining applications. High-dimensional spaces are sparse (Bellman, R.E., Adaptive Control Processes, Princeton University Press, Princeton (1961)), counter-intuitive (Wegman, E.,J. Ann. Statist. 41:457-471 (1970)), and inherently difficult to understand, and their structure cannot be easily extracted with conventional graphical techniques. However, experience has shown that, regardless of origin, most multivariate data in R^d are almost never truly *d*-dimensional. That is, the underlying structure of the data is almost always of dimensionality lower than *d*. Extracting that structure into a low-dimensional representation has been the

subject of countless studies over the past 50 years, and several techniques have been devised and popularized through the widespread availability of commercial statistical software. These techniques are divided into two main categories: linear and nonlinear.

[0003] Perhaps the most common linear dimensionality reduction technique is principal component analysis, or PCA (Hotelling, H., J. Edu. Psychol. 24:417-441; 498-520 (1933)). PCA reduces a set of partially cross-correlated data into a smaller set of orthogonal variables with minimal loss in the contribution to variation. The method has been extensively tested and is well-understood, and several effective algorithms exist for computing the projection, ranging from singular value decomposition to neural networks (Oja, E., Subspace Methods of Pattern Recognition, Research Studies Press, Letchworth, England (1983); Oja, E., Neural Networks 5:927-935 (1992); Rubner, J., and Tavan, P., Europhys. Lett. 10:693-698 (1989)). PCA makes no assumptions about the probability distributions of the original variables, but is sensitive to outliers, missing data, and poor correlations due to poorly distributed variables. More importantly, the method cannot deal effectively with nonlinear structures, curved manifolds, and arbitrarily shaped clusters.

[0004] A more general methodology is Friedman's exploratory projection pursuit (EPP) (Friedman, J.H., and Tukey, J.W., IEEE Trans. Computers 23:881-890 (1974); Friedman, J.H., J. Am. Stat. Assoc. 82:249-266 (1987)). This method searches multidimensional data sets for interesting projections or views. The "interestingness" of a projection is typically formulated as an index, and is numerically maximized over all possible projections of the multivariate data. In most cases, projection pursuit aims at identifying views that exhibit significant clustering and reveal as much of the non-normally distributed structure in the data as possible. The method is general, and includes several well-known linear projection techniques as special cases, including principal component analysis (in this case, the index of interestingness is simply the sample variance of the projection). Once an

interesting projection has been identified, the structure that makes the projection interesting may be removed from the data, and the process can be repeated to reveal additional structure. Although projection pursuit attempts to express some nonlinearities, if the data set is high-dimensional and highly nonlinear it may be difficult to visualize it with linear projections onto a low-dimensional display plane, even if the projection angle is carefully chosen.

[0005] Several approaches have been proposed for reproducing the nonlinear structure of higher-dimensional data spaces. The best-known techniques are self-organizing maps, auto-associative neural networks, multidimensional scaling, and nonlinear mapping.

[0006] Self-organizing maps or Kohonen networks (Kohonen, T., Self-Organizing Maps, Springer-Verlag, Heidelberg (1996)) were introduced by Kohonen in an attempt to model intelligent information processing, i.e. the ability of the brain to form reduced representations of the most relevant facts without loss of information about their interrelationships. Kohonen networks belong to a class of neural networks known as competitive learning or self-organizing networks. Their objective is to map a set of vectorial samples onto a two-dimensional lattice in a way that preserves the topology and density of the original data space. The lattice points represent neurons which receive identical input, and compete in their activities by means of lateral interactions. The main application of self-organizing maps is in visualizing complex multi-variate data on a 2-dimensional plot, and in creating abstractions reminiscent of these obtained from clustering methodologies. These reduced representations can subsequently be used for a variety of pattern recognition and classification tasks.

[0007] Another methodology is that of auto-associative neural networks (DeMers, D., and Cottrell, G., Adv. Neural Info. Proces. Sys. 5:580-587 (1993); Garrido, L., et al., Int. J. Neural Sys. 6:273-282 (1995)). These are multi-layer feed-forward networks trained to reproduce their inputs as desired outputs. They consist of an input and an output layer containing as many

neurons as the number of input dimensions, and a series of hidden layers having a smaller number of units. In the first part of the network, each sample is reorganized, mixed, and compressed into a compact representation encoded by the middle layer. This representation is then decompressed by the second part of the network to reproduce the original input. Auto-associative networks can be trained using conventional back-propagation or any other related technique available for standard feed-forward architectures. A special version of the multilayer perceptron, known as a replicator network (Hecht-Nielsen, R., Science 269:1860-1863 (1995)), has been shown to be capable of representing its inputs in terms of their "natural coordinates". These correspond to coordinates in an m-dimensional unit cube that has been transformed elastically to fit the distribution of the data. Although in practice it may be difficult to determine the inherent dimensionality of the data, the method could, in theory, be used for dimensionality reduction using a small value of m.

[0008] The aforementioned techniques can be used only for dimension reduction. A more broadly applicable method is multidimensional scaling (MDS) or nonlinear mapping (NLM). This approach emerged from the need to visualize a set of objects described by means of a similarity or distance matrix. The technique originated in the field of mathematical psychology (see Torgeson, W. S., Psychometrika, 1952, and Kruskal, J. B. Phychometrika, 1964, both of which are incorporated by reference in their entirety), and has two primary applications: 1) reducing the dimensionality of high-dimensional data in a way that preserves the original relationships of the data objects, and 2) producing Cartesian coordinate vectors from data supplied directly in the form of similarities or proximities, so that they can be analyzed with conventional statistical and data mining techniques.

[0009] Given a set of k objects, a symmetric matrix, r_{ij}, of relationships between these objects, and a set of images on a m-dimensional display plane $\{y_i, i = 1, 2, .., k; y_i \in \Re^m\}$, the problem is to place y_i onto the plane in such a

way that their Euclidean distances $d_{ij} = ||y_i - y_j||$ approximate as closely as possible the corresponding values r_{ij}. The quality of the projection is determined using a loss function such as Kruskal's stress:

$$S = \sqrt{\frac{\sum_{i<j}(d_{ij} - r_{ij})^2}{\sum_{i<j} r_{ij}^2}} \qquad (1)$$

which is numerically minimized in order to find the optimal configuration. The actual embedding is carried out in an iterative fashion by: 1) generating an initial set of coordinates y_i, 2) computing the distances d_{ij}, 3) finding a new set of coordinates y_i using a steepest descent algorithm such as Kruskal's linear regression or Guttman's rank-image permutation, and 4) repeating steps 2 and 3 until the change in the stress function falls below some predefined threshold.

[0010] A particularly popular implementation is Sammon's nonlinear mapping algorithm (Sammon, J. W. IEEE Trans. Comp., 1969). This method uses a modified stress function:

$$E = \frac{\sum_{i<j}^{k} \frac{[r_{ij} - d_{ij}]^2}{r_{ij}}}{\sum_{i<j}^{k} r_{ij}} \qquad (2)$$

which is minimized using steepest descent. The initial coordinates, y_i, are determined at random or by some other projection technique such as principal componenet analysis, and are updated using Eq. 3:

$$y_{ij}(t+1) = y_{ij}(t) - \lambda \Delta_{ij}(t) \qquad (3)$$

where t is the iteration number and λ is the learning rate parameter, and

$$\Delta_{ij}(t) = \frac{\frac{\partial E(t)}{\partial y_{ij}(t)}}{\left|\frac{\partial^2 E(t)}{\partial y_{ij}(t)^2}\right|} \qquad (4)$$

[0011] There is a wide variety of MDS algorithms involving different error functions and optimization heuristics, which are reviewed in Schiffman,

Reynolds and Young, *Introduction to Multidimensional Scaling*, Academic Press, New York (1981); Young and Hamer, *Multidimensional Scaling: History, Theory and Applications*, Erlbaum Associates, Inc., Hillsdale, NJ (1987); Cox and Cox, *Multidimensional Scaling*, Number 59 in *Monographs in Statistics and Applied Probability*, Chapman-Hall (1994), and Borg, I., Groenen, P., Modern Multidimensional Scaling, Springer-Verlag, New York, (1997). The contents of these publications are incorporated herein by reference in their entireties. Different forms of NLM will be discussed in greater detail below.

[0012] Unfortunately, the quadratic nature of the stress function (Eqs. 1 and 2, and their variants) make these algorithms impractical for large data sets containing more than a few hundred to a few thousand items. Several attempts have been devised to reduce the complexity of the task. Chang and Lee (Chang, C. L., and Lee, R. C. T., *IEEE Trans. Syst., Man, Cybern.*, 1973, *SMC-3*, 197-200) proposed a heuristic relaxation approach in which a subject of the original objects (the frame) are scaled using a Sammon-like methodology, and the remaining objects are then added to the map by adjusting their distances to the objects in the frame. An alternative approach proposed by Pykett (Pykett, C. E., *Electron. Lett.*, 1978, *14*, 799-800) is to partition the data into a set of disjoint clusters, and map only the cluster prototypes, i.e. the centroids of the pattern vectors in each class. In the resulting two-dimensional plots, the cluster prototypes are represented as circles whose radii are proportional to the spread in their respective classes. Lee, Slagle and Blum (Lee, R. C. Y., Slagle, J. R., and Blum, H., *IEEE Trans. Comput.*, 1977, *C-27*, 288-292) proposed a triangulation method which restricts attention to only a subset of the distances between the data samples. This method positions each pattern on the plane in a way that preserves its distances from the two nearest neighbors already mapped. An arbitrarily selected reference pattern may also be used to ensure that the resulting map is globally ordered. Biswas, Jain and Dubes (Biswas, G., Jain, A. K., and Dubes,

R. C., *IEEE Trans. Pattern Anal. Machine Intell.*, 1981, *PAMI-3(6)*, 701-708) later proposed a hybrid approach which combined the ability of Sammon's algorithm to preserve global information with the efficiency of Lee's triangulation method. While the triangulation can be computed quickly compared to conventional MDS methods, it tries to preserve only a small fraction of relationships, and the projection may be difficult to interpret for large data sets.

[0013] The methods described above are iterative in nature, and do not provide an explicit mapping function that can be used to project new, unseen patterns in an efficient manner. The first attempt to encode a nonlinear mapping as an explicit function is due to Mao and Jain (Mao, J., and Jain, A.K., *IEEE Trans. Neural Networks* 6(2):296-317 (1995)). They proposed a 3-layer feed-forward neural network with n input and m output units, where n and m are the number of input and output dimensions, respectively. The system is trained using a special back-propagation rule that relies on errors that are functions of the inter-pattern distances. However, because only a single distance is examined during each iteration, these networks require a very large number of iterations and converge extremely slowly.

[0014] An alternative methodology is to employ Sammon's nonlinear mapping algorithm to project a small random sample of objects from a given population, and then "learn" the underlying nonlinear transform using a multilayer neural network trained with the standard error back-propagation algorithm or some other equivalent technique (see for example, Haykin, S. Neural Networks: A Comprehensive Foundation. Prentice-Hall, 1998). Once trained, the neural network can be used in a feed-forward manner to project the remaining objects in the plurality of objects, as well as new, unseen objects. Thus, for a nonlinear projection from n to m dimensions, a standard 3-layer neural network with n input and m output units is used. Each n-dimensional object is presented to the input layer, and its coordinates on the m-dimensional nonlinear map are obtained by the respective units in the

output layer (Pal, N. R. Eluri, V. K., *IEEE Trans. Neural Net.*, 1142-1154 (1998)).

[0015] The distinct advantage of this approach is that it captures the nonlinear mapping relationship in an explicit function, and allows the scaling of additional patterns as they become available, without the need to reconstruct the entire map. It does, however, rely on conventional MDS methodologies to construct the nonlinear map of the training set, and therefore the method is inherently limited to relatively small samples.

[0016] Hence there is a need for a method that can efficiently process large data sets, e.g., data sets containing hundreds of thousands to millions of items. Moreover, just like Mao and Jain (Mao, J., and Jain, A.K., *IEEE Trans. Neural Networks* 6(2):296-317 (1995)) and Pal and Eluri (Pal, N. R. Eluri, V. K., *IEEE Trans. Neural Net.*, 1142-1154 (1998)), a method is needed that is incremental in nature, and allows the mapping of new samples as they become available, without the need to reconstruct an entire map.

SUMMARY OF THE INVENTION

[0017] A method and computer product is presented for mapping input patterns of high dimensionality into a lower dimensional space so as to preserve the relationships between these patterns in the higher dimensional space. A subset of the input patterns is chosen and mapped into the lower dimensional space using an iterative process based on subset refinements. A set of local regions is defined using a clustering methodology, and a local neural network is associated with each of these regions, and trained in accordance with the mapping obtained from the iterative process. Additional input patterns not in the subset are mapped into the lower dimensional space by using one of the local neural networks. In an alternative embodiment, the local neural networks are only used after training and use of a global neural network. The global neural network is trained in accordance with the results of the mapping produced by the iterative process. Input patterns are fed into

the global neural network, resulting in patterns in the lower dimensional space. Local neural networks are then used to refine the results of the global network.

[0018] The method and computer product described herein permits the mapping of massive data sets from a higher dimensional space to a lower dimensional space. Moreover, the method allows the mapping of new input patterns as they become available, without the need to reconstruct an entire map.

[0019] The foregoing and other features and advantages of the invention will be apparent from the following, more particular description of a preferred embodiment of the invention, as illustrated in the accompanying drawings.

BRIEF DESCRIPTION OF THE DRAWINGS/FIGURES

[0020] FIG. 1 illustrates possibilities for a single hypothetical pairwise relationship and distances of corresponding objects on a nonlinear map.

[0021] FIG. 2 is a flowchart illustrating the phases of the method of the invention.

[0022] FIG. 3 is a flowchart illustrating the training phase of the invention, according to an embodiment.

[0023] FIG. 4 is a flowchart illustrating the use of a fuzzy clustering methodology in the selection of reference patterns, according to an embodiment of the invention.

[0024] FIG. 5 illustrates the concept of Voronoi cells, as used in an embodiment of the invention.

[0025] FIG. 6 is a flowchart illustrating the projection of input patterns, according to an embodiment of the invention.

[0026] FIG. 7 illustrates the operation of local neural networks, according to an embodiment of the invention.

[0027] FIG. 8 is a flowchart illustrating the training phase of the invention, according to an alternative embodiment.

[0028] FIG. 9 is a flowchart illustrating the projection of input patterns, according to an alternative embodiment of the invention.

[0029] FIG. 10 illustrates the operation of global and local neural networks, according to an alternative embodiment of the invention.

[0030] FIG. 11 illustrates a computing environment within which the invention can operate.

DETAILED DESCRIPTION OF THE INVENTION

[0031] A preferred embodiment of the present invention is now described with reference to the figures where like reference numbers indicate identical or functionally similar elements. Also in the figures, the left most digit of each reference number corresponds to the figure in which the reference number is first used. While specific configurations and arrangements are discussed, it should be understood that this is done for illustrative purposes only. A person skilled in the relevant art will recognize that other configurations and arrangements can be used without departing from the spirit and scope of the invention. It will be apparent to a person skilled in the relevant art that this invention can also be employed in a variety of other devices and applications.

I. Introduction

A. Overview

[0032] A neural network architecture for reducing the dimensionality of very large data sets is presented here. The method is rooted on the principle of probability sampling, i.e. the notion that a small number of randomly chosen members of a given population will tend to have the same characteristics, and in the same proportion, with the population as a whole. The approach employs an iterative algorithm based on subset refinements to nonlinearly map a small random sample which reflects the overall structure of the data, and then "learns" the underlying nonlinear transform using a set of distributed neural

networks, each specializing in a particular domain of the feature space. The partitioning of the data space can be carried out using a clustering methodology. This local approach eliminates a significant portion of the imperfection of the nonlinear maps produced by a single multi-layer perceptron, and does so without a significant computational overhead. The proposed architecture is general and can be used to extract constraint surfaces of any desired dimensionality.

[0033] The following section discusses methods that can be used to nonlinearly map a random subset of the data.

B. Nonlinear Mapping Using Subset Refinements

1. Overview

[0034] A nonlinear mapping algorithm that is well suited for large data sets is presented in U.S. Patent Application 09/303,671, filed May 3, 1999, titled, "Method, System and Computer Program Product for Nonlinear Mapping of Multidimensional Data", and U.S. Patent Application 09/073,845, filed May 7, 1998, titled, "Method, System and Computer Program Product for Representing Proximity Data in a Multidimensional Space". This approach is to use iterative refinement of coordinates based on partial or stochastic errors.

[0035] The method uses a self-organizing principle to iteratively refine an initial (random or partially ordered) configuration of objects by analyzing only a subset of objects and their associated relationships at a time. The relationship data may be complete or incomplete (i.e. some relationships between objects may not be known), exact or inexact (i.e. some or all relationships may be given in terms of allowed ranges or limits), symmetric or asymmetric (i.e. the relationship of object A to object B may not be the same as the relationship of B to A) and may contain systematic or stochastic errors.

[0036] The relationships between objects may be derived directly from observation, measurement, a priori knowledge, or intuition, or may be

determined directly or indirectly using any suitable technique for deriving such relationships.

[0037] The invention determines the coordinates of a plurality of objects on the *m*-dimensional nonlinear map by:

(1) placing the objects on the *m*-dimensional nonlinear map;

(2) selecting a subset of the objects, wherein the selected subset of objects includes associated relationships between objects in the selected subset;

(3) revising the coordinate(s) of one or more objects in the selected subset of objects on the *m*-dimensional nonlinear map based on the relationship(s) between some of these objects and their corresponding distance(s) on the nonlinear map;

(4) repeating steps (2) and (3) for additional subsets of objects from the plurality of objects.

[0038] In one embodiment, subsets of objects can be selected randomly, semi-randomly, systematically, partially systematically, etc. As subsets of objects are analyzed and their distances on the nonlinear map are revised, the set of objects tends to self-organize.

[0039] In a preferred embodiment, the invention iteratively analyzes a pair of objects at a time, that is, step (2) is carried out by selecting a pair of objects having an associated pairwise relationship. Pairs of objects can be selected randomly, semi-randomly, systematically, partially systematically, etc. Novel algorithms and techniques for pairwise analysis are provided in the sections below. This embodiment is described for illustrative purposes only and is not limiting.

2. Pairwise Relationship Matrices without Uncertainties

a. Full Pairwise Relationship Matrices without Uncertainties

[0040] The discussion in this section assumes that all pairwise relationships are known, and they are all exact. In a preferred embodiment, the method

starts with an initial configuration of points generated at random or by some other procedure such as principal component analysis. This initial configuration is then continuously refined by repeatedly selecting two objects, i, j, at random, and modifying their coordinates on the nonlinear map according to Eq. 5:

$$y_i(t+1) = f(t, y_i(t), y_j(t), r_{ij}) \tag{5}$$

where t is the current iteration, $y_i(t)$ and $y_j(t)$ are the current coordinates of the i-th and j-th objects on the nonlinear map, $y_i(t+1)$ are the new coordinates of the i-th object on the nonlinear map, and r_{ij} is the relationship between the i-th and j-th objects. $f(.)$ in Eq. 5 above can assume any functional form. Ideally, this function should try to minimize the difference between the distance on the nonlinear map and the actual relationship between the i-th and j-th objects. For example, $f(.)$ may be given by Eq. 6:

$$y_i(t+1) = 0.5\lambda(t)\frac{r_{ij} - d_{ij}(t)}{d_{ij}(t)}(y_i(t) - y_j(t)) \tag{6}$$

where t is the iteration number, $d_{ij} = \|y_i(t) - y_j(t)\|$, and $\lambda(t)$ is an adjustable parameter, referred to hereafter as the "learning rate". This process is repeated for a fixed number of cycles, or until some global error criterion is minimized within some prescribed tolerance. A large number of iterations are typically required to achieve statistical accuracy.

[0041] The method described above is generally reminiscent of the error back-propagation procedure for training artificial neural networks described in Werbos, Beyond Regression: New Tools for Prediction and Analysis in the Behavioral Sciences, PhD Thesis, Harvard University, Cambridge, MA (1974), and Rumelhart and McClelland, Eds., Parallel Distributed Processing: Explorations in the Microstructure of Cognition, Vol. 1, MIT Press, Cambridge, MA (1986), both of which are incorporated herein by reference in their entireties.

[0042] The learning rate $\lambda(t)$ in EQ. 6 plays a key role in ensuring convergence. If λ is too small, the coordinate updates are small, and

convergence is slow. If, on the other hand, λ is too large, the rate of learning may be accelerated, but the nonlinear map may become unstable (i.e. oscillatory). Typically, λ ranges in the interval [0, 1] and may be fixed, or it may decrease monotonically during the refinement process. Moreover, λ may also be a function of i, j, r_{ij}, and/or d_{ij}, and can be used to apply different weights to certain objects, relationships, distances and/or relationship or distance pairs. For example, λ may be computed by Eq. 7:

$$\lambda(t) = (\lambda_{\min} + t\frac{\lambda_{\max} - \lambda_{\min}}{T})\frac{1}{1 + ar_{ij}} \tag{7}$$

or Eq. 8:

$$\lambda(t) = (\lambda_{\min} + t\frac{\lambda_{\max} - \lambda_{\min}}{T})e^{-ar_{ij}} \tag{8}$$

where λ_{max} and λ_{min} are the (unweighted) starting and ending learning rates such that λ_{max}, $\lambda_{min} \in [0,1]$; T is the total number of refinement steps (iterations), t is the current iteration number, and α is a constant scaling factor. EQ. 7 and 8 have the effect of decreasing the correction at large separations (weak relationships), thus creating a nonlinear map which preserves strong relationships (short distances) more faithfully than weak ones. Weighting is discussed in greater detail below.

[0043] One of the main advantages of this approach is that it makes partial refinements possible. It is often sufficient that the pairwise similarities are represented only approximately to reveal the general structure and topology of the data. Unlike traditional MDS, this approach allows very fine control of the refinement process. Moreover, as the nonlinear map self-organizes, the pairwise refinements become cooperative, which partially alleviates the quadratic nature of the problem.

[0044] The embedding procedure described above does not guarantee convergence to the global minimum (i.e., the most faithful embedding in a least-squares sense). If so desired, the refinement process may be repeated a

number of times from different starting configurations and/or random number seeds.

[0045] The general algorithm described above can also be applied when the pairwise similarity matrix is incomplete, i.e. when some of the pairwise similarities are unknown, when some of the pairwise similarities are uncertain or corrupt, or both of the above. These cases are discussed separately below.

b. Sparse Pairwise Relationship Matrices without Uncertainties

[0046] The general algorithm described above can also be applied when the pairwise relationship matrix is incomplete, i.e. when some of the pairwise relationships are unknown. In this case, a similar algorithm to the one described above can be used, with the exception that the algorithm iterates over pairs of objects for which the relationships are known. In this case, the algorithm identifies configurations in space that satisfy the known pairwise relationships; the unknown pairwise relationships adapt during the course of refinement and eventually assume values that lead to a satisfactory embedding of the known relationships.

[0047] Depending on the number of missing data, there may be more than one satisfactory embeddings (mappings) of the original relationship matrix. In this case, different configurations (maps) may be derived from different starting configurations or random number seeds. In some applications such as searching the conformational space of molecules, this feature provides a significant advantage over some alternative techniques. All variants of the original algorithm (see Sections below) can be used in this context.

3. Pairwise Relationship Matrices with Bounded Uncertainties

[0048] The general algorithm described above can also be applied when the pairwise relationships contain bounded uncertainties, i.e. when some of the pairwise relationships are only known to within certain fixed tolerances (for

example, the relationships are known to lie within a range or set of ranges with prescribed upper and lower bounds). In this case, a similar algorithm to the one described above can be used, with the exception that the distances on the nonlinear map are corrected only when the corresponding objects lie outside the prescribed bounds. For example, assume that the relationship between two objects, i and j, is given in terms of an upper and lower bound, r_{max} and r_{min}, respectively. When this pair of objects is selected during the course of the refinement, the distance of the objects on the nonlinear map is computed, and denoted as d_{ij}. If d_{ij} is larger than r_{max}, the coordinates of the objects are updated using r_{max} as the target distance (Eq. 9):

$$\mathbf{y}_i(t+1) = f(t, \mathbf{y}_i(t), \mathbf{y}_j(t), r_{\max}) \tag{9}$$

[0049] Conversely, if d_{ij} is smaller than r_{min}, the coordinates of the objects are updated using r_{min} as the target distance (Eq. 10):

$$\mathbf{y}_i(t+1) = f(t, \mathbf{y}_i(t), \mathbf{y}_j(t), r_{\min}) \tag{10}$$

If d_{ij} lies between the upper and lower bounds (i.e. if $r_{min} \neq d_{ij} \neq r_{max}$), no correction is made. In other words, the algorithm attempts to match the upper bound if the current distance between the objects is greater than the upper bound, or the lower bound if the current distance between the objects is lower than the lower bound. If the distance between the objects lies within the upper and lower bounds, no correction is made.

[0050] This algorithm can be extended in the case where some of the pairwise relationships are given by a finite set of allowed discrete values, or by a set of ranges of values, or some combination thereof. For the purposes of the discussion below, we consider discrete values as ranges of zero width (e.g. the discrete value of 2 can be represented as the range [2,2]).

[0051] Various possibilities for a single hypothetical pairwise relationship and the current distance of the corresponding objects on the nonlinear map are illustrated in FIG. 1, where shaded areas 110, 112 and 114 denote allowed ranges for a given pairwise relationship. Distances d1-d5 illustrate 5 different possibilities for the current distance between the corresponding objects on the

nonlinear map. Arrows 116, 118, 120 and 122 indicate the direction of the correction that should be applied on the objects on the map. Arrows 118 and 122 point to the left, indicating that the coordinates of the associated objects on the nonlinear map should be updated so that the objects come closer together. Arrows 116 and 120 point to the right, indicating that the coordinates of the associated objects should be updated so that the objects become more distant.

[0052] As in the case of a single range, if the current distance of a selected pair of objects on the nonlinear map lies within any of the prescribed ranges, no coordinate update takes place (i.e., case d1 in FIG. 1). If not, the correction is applied using the nearest range boundary as the target distance (i.e., cases d2-d5 in FIG. 1). For example, if the relationship between a given pair of objects lies in the ranges [1,2], [3,5] and [6,7] and their current distance on the nonlinear map is 2.9 (d5 in FIG. 1), the correction takes place using 3 as the target distance (r_{ij}) in Eq. 5. If, however, the current distance is 2.1, the coordinates are updated using 2 as the target distance (r_{ij}) in Eq. 5.

[0053] This deterministic criterion may be replaced by a stochastic or probabilistic one in which the target distance is selected either randomly or with a probability that depends on the difference between the current distance and the two nearest range boundaries. In the example described above (d5 in FIG. 1), a probabilistic choice between 2 and 3 as a target distance could be made, with probabilities of, for example, 0.1 and 0.9, respectively (that is, 2 could be selected as the target distance with probability 0.1, and 3 with probability 0.9). Any method for deriving such probabilities can be used. Alternatively, either 2 or 3 could be chosen as the target distance at random.

[0054] For example, bounded uncertainties in the pairwise relationships may represent stochastic or systematic errors or noise associated with a physical measurement, and can, in general, differ from one pairwise relationship to another. A typical example are the Nuclear Overhauser Effects (NOE's) in multidimensional Nuclear Magnetic Resonance spectrometry. Alternatively,

the uncertainty may result from multiple measurements of a given relationship.

[0055] An alternative algorithm for dealing with uncertainties is to reduce the magnitude of the correction for pairs of objects whose relationship is thought to be uncertain. In this scheme, the magnitude of the correction, as determined by the learning rate in Eq. 8, for example, is reduced for pairwise relationships which are thought to be uncertain. The magnitude of the correction may depend on the degree of uncertainty associated with the corresponding pairwise relationship (for example, the magnitude of the correction may be inversely proportional to the uncertainty associated with the corresponding pairwise relationship). If the existence and/or magnitude of the errors is unknown, then the errors can be determined automatically by the algorithm.

4. Pairwise Relationship Matrices with Unbounded Uncertainties

The ideas described in the preceding Sections can be applied when some of the pairwise relationships are thought to contain corrupt data, that is when some of the pairwise relationships are incorrect and bear essentially no relationship to the actual values. In this case, "problematic" relationships can be detected during the course of the algorithm, and removed from subsequent processing. In other words, the objective is to identify the corrupt entries and remove them from the relationship matrix. This process results in a sparse relationship matrix, which can be refined using the algorithm in Section 2.a above.

5. Modifications of the Basic Algorithm

[0056] In many cases, the algorithm described above may be accelerated by pre-ordering the data using a suitable statistical method. For example, if the proximities are derived from data that is available in vectorial or binary form, the initial configuration of the points on the nonlinear map may be computed using principal component analysis. In a preferred embodiment, the initial

configuration may be constructed from the first m principal components of the feature matrix (i.e. the m latent variables which account for most of the variance in the data). This technique can have a profound impact on the speed of refinement. Indeed, if a random initial configuration is used, a significant portion of the training time is spent establishing the general structure and topology of the nonlinear map, which is typically characterized by large rearrangements. If, on the other hand, the input configuration is partially ordered, the error criterion can be reduced relatively rapidly to an acceptable level.

[0057] If the data is highly clustered, by virtue of the sampling process low-density areas may be refined less effectively than high-density areas. In one embodiment, this tendency may be partially compensated by a modification to the original algorithm, which increases the sampling probability in low-density areas. This type of biased sampling may be followed with regular, unbiased sampling, and this process may be repeated any number of times in any desired sequence.

[0058] Generally, the basic algorithm does not distinguish weak from strong relationships (short-range and long-range distances, respectively). One method to ensure that strong relationships are preserved more faithfully than weak relationships is to weight the coordinate update in EQ. 5 (or, equivalently, the learning rate λ in EQ. 7 and 8) by a scaling factor that is inversely proportional to the.strength (magnitude) of the relationship

[0059] An alternative (and complementary) approach is to ensure that objects at close separation are sampled more extensively than objects at long separation. For example, an alternating sequence of global and local refinement cycles, similar to the one described above, can be employed. In this embodiment, a phase of global refinement is initially carried out, after which, the resulting nonlinear map is partitioned into a regular grid. The points (objects) in each cell of the grid are then subjected to a phase of local refinement (i.e. only objects from within the same cell are compared and

refined). Preferably, the number of sampling steps in each cell should be proportional to the number of objects contained in that cell. This process is highly parallelizable. This local refinement phase is then followed by another global refinement phase, and the process is repeated for a prescribed number of cycles, or until the embedding error is minimized within a prescribed tolerance. Alternatively, the grid method may be replaced by another suitable method for identifying proximal points, such as clustering, for example.

[0060] The methods described herein may be used for incremental mapping. That is, starting from an organized nonlinear map of a set of objects, a new set of objects may be added without modification of the original map. In an exemplary embodiment, the new set of objects may be "diffused" into the existing map, using a modification of the basic algorithm described above. In particular, Eq. 5 and 6 can be used to update only the additional objects. In addition, the sampling procedure ensures that the selected pairs contain at least one object from the new set. That is, two objects are selected at random so that at least one of these objects belongs to the new set. Alternatively, each new object may be added independently using the approach described above.

II. Method

A. Nonlinear Mapping Networks – Algorithm I

[0061] The process described herein uses the iterative nonlinear mapping algorithm described in Section II to multidimensionally scale a small random sample of a set of input patterns of dimensionality n, and then "learns" the underlying nonlinear transform using an artificial neural network. For a nonlinear projection from n to m dimensions, a simple 3-layer network with n input and m output units can be employed. The network is trained to reproduce the input/output coordinates produced by the iterative algorithm, and thus encodes the mapping in its synaptic parameters in a compact, analytical manner. Once trained, the neural network can be used in a feed-

forward fashion to project the remaining members of the input set, as well as new, unseen samples with minimal distortion.

[0062] The method of the invention is illustrated generally in FIG. 2. The method begins at step 205. In step 210, the training of a neural network takes place, where the training is based on the results (i.e., the inputs and outputs) of the iterative algorithm. In step 215, points in R^n are projected into R^m by a feed-forward pass through the trained neural network The process concludes with step 220.

B. Local Nonlinear Mapping Networks – Algorithm II

[0063] The embodiment of the invention described in this section represents a variation of the above algorithm. This approach is based on local learning. Instead of using a single "global" network to perform the nonlinear mapping across the entire input data space R^n, this embodiment partitions the space into a set of Voronoi polyhedra, and uses a separate "local" network to project the patterns in each partition. Given a set of reference points $P = \{\mathbf{p}_1, \mathbf{p}_2, \ldots\}$ in R^n, a Voronoi polyhedron (or Voronoi cell), $v(\mathbf{p})$, is a convex polytope associated with each reference point p which contains all the points in R^n that are closer to $\mathbf{p}$ than any other point in P:

$$v(\mathbf{p}) = \{\mathbf{x} \in R^n \mid d(\mathbf{x},\mathbf{p}) \leq d(\mathbf{x},\mathbf{q}) \quad \forall \mathbf{p},\mathbf{q} \in P, \mathbf{p} \neq \mathbf{q}\} \tag{11}$$

where d() is a distance function. In an embodiment of the invention, d() is the Euclidean distance function. Voronoi cells partition the input data space R^n into local regions "centered" at the reference points P, also referred to as centroids. Hereafter, the local networks associated with each Voronoi cell are said to be centered at the points P, and the distance of a point in R^n from a local network will refer to the distance of that point from the network's center.

[0064] The training phase involves the following general steps: a training set is extracted from the set of input patterns and mapped using the iterative nonlinear mapping algorithm described in Section II. A set of reference points in the input space R^n is then selected, and the objects comprising the training

set are partitioned into disjoint sets containing the patterns falling within the respective Voronoi cells. Patterns that lie on the sides and vertices of the Voronoi cells (i.e. are equidistant to two or more points in *P*), are arbitrarily assigned to one of the cells. A local network is then assigned to each cell, and is trained to reproduce the input/output mapping of the input patterns in that cell. While the direct nonlinear map is obtained globally, the networks are trained locally using only the input patterns within their respective Voronoi partitions. Again, simple 3-layer perceptrons with *n* input and *m* output units can be employed, where *n* and *m* are the dimensionalities of the input and output spaces, respectively.

[0065] The training phase of the method of the invention therefore involves the following steps as illustrated in FIG. 3. The training phase begins at step 305. In step 310, a random set of points $\{\mathbf{x}_i, i=1,2...,k; \mathbf{x}_i \in R^n\}$ is extracted from the set of input patterns. In step 315, the points $\mathbf{x}_i$ are mapped from R^n to R^m using the iterative nonlinear mapping algorithm described in Section II ($\mathbf{x}_i \rightarrow \mathbf{y}_i$, i = 1,2,...,k, $\mathbf{x}_i \in R^n$, $\mathbf{y}_i \in R^m$). This mapping serves to define a training set *T* of ordered pairs $(\mathbf{x}_i, \mathbf{y}_i)$, $T = \{(\mathbf{x}_i, \mathbf{y}_i), i = 1,2,...,k\}$.

[0066] In step 320, a set of reference points $P = \{\mathbf{c}_i, i = 1,2,...c; \mathbf{c}_i \in R^n\}$ is determined. In an embodiment of the invention, the reference points $\mathbf{c}_i$ are determined using a clustering algorithm described in greater detail below. In step 325, the training set T is partitioned into *c* disjoint clusters based on the distance of each point $\mathbf{x}_i$ from each reference point. The set of disjoint clusters is denoted $\{C_j = \{(\mathbf{x}_i, \mathbf{y}_i): d(\mathbf{x}_i, \mathbf{c}_j) \leq d(\mathbf{x}_i, \mathbf{c}_k)\}$ for all $k \neq j$; j = 1,2,...,*c*; i = 1,2,...,*k*$\}\}$. In step 330, *c* independent local networks $\{Net_i^L, i = 1,2,...,c\}$ are trained with the respective training subsets C_i derived in step 325. The training phase concludes with step 335.

[0067] Clearly, an important choice to be made concerns the partitioning. In general, the reference points $\mathbf{c}_i$ (determined in step 320) should be well distributed and should produce balanced partitions that contain a comparable number of training patterns. This is necessary in order to avoid the creation of

poorly optimized networks due to an insufficient number of training cases. In one embodiment, described here, the reference points $\mathbf{c}_i$ can be determined using the fuzzy clustering means (FCM) algorithm (Bezdek, J.C., Pattern Recognition with Fuzzy Objective Function Algorithms. Plenum Press, 1981). The FCM algorithm uses the probabilistic constraint that the sum of the memberships of a data point over all clusters must be equal to 1, and has been most successful in situations where the final objective is a crisp decision, as is the case in the problem at hand.

[0068] The FCM algorithm attempts to minimize the objective function:

$$J_q = \sum_{j=1}^{C} \sum_{i=1}^{N} m_{ij}^q \, d^2\left(\mathrm{x}_i, \mathrm{c}_j\right) \tag{12}$$

over a set of points $\mathbf{x}_i$, i = 1, 2, ...N, with respect to the fuzzy degrees of membership m_{ij}, the "fuzziness" index q, and the cluster centroids $\mathbf{c}_j$, j = 1, 2, ...C, where C is the total number of clusters, and m_{ij} is the degree of membership of the i-th pattern to the j-th cluster. In addition, the solution must satisfy the constraints that the membership value of each pattern in any given cluster must be less than or equal to one:

$$0 \le m_{ij} \le 1 \tag{13}$$

and the sum of its membership values over all clusters must be equal to 1:

$$\sum_{j=1}^{C} m_{ij} = 1 \tag{14}$$

[0069] In an embodiment of the invention, the squared distance $d^2(\mathbf{x}_i, \mathbf{c}_j)$ is the Euclidean distance between the *i*-th point in R^n and the *j*-th reference point. Equations 13 and 14 ensure that the solutions to Eq. 6 represent true fuzzy partitions of the data set among all the specified classes. In the above equation, $q \in [1, \infty)$ is a weighting exponent known as the "fuzziness index" or "fuzzifier" that controls the degree of fuzziness of the resulting clusters. For q = 1 the partitions are crisp, and as $q \rightarrow \infty$ the clusters become increasingly fuzzy.

[0070] The determination of reference points $\mathbf{c}_j$, j = 1, 2, ...C using FCM is illustrated in FIG. 4, according to an embodiment of the invention. The process starts with step 405. In step 410, an iteration counter *p* is initialized. In step 415, an initial choice for the cluster centroids $\{\mathbf{c}_j,\ j = 1, 2, ...C\}$ is made. Given this choice for $\{\mathbf{c}_i\}$, in step 420 the degree of membership of each point $\mathbf{x}_i$ in each cluster is calculated using the following formula:

$$m_{ij} = \frac{\left[\frac{1}{d^2(\mathbf{x}_i,\mathbf{c}_j)}\right]^{\frac{1}{q-1}}}{\sum_{k=1}^{C}\left[\frac{1}{d^2(\mathbf{x}_i,\mathbf{c}_k)}\right]^{\frac{1}{q-1}}} \tag{15}$$

In step 425, the objective function J_q^p is evaluated using Eq. 16:

$$J_q^p = \sum_{j=1}^{C}\sum_{i=1}^{N} m_{ij}^{\,q}\, d^2(\mathbf{x}_i,\mathbf{c}_j) \tag{16}$$

In step 430, new centroids $\mathbf{c}_j$ are computed using the formula:

$$\mathbf{c}_j = \frac{\sum_{i=1}^{N} m_{ij}^{\,q}\mathbf{x}_i}{\sum_{i=1}^{N} m_{ij}^{\,q}} \tag{17}$$

In step 435, m_{ij} are recalculated using the new centroids and Eq. 15. In step 440, the iteration counter *p* is incremented. In step 445, J_q^p is recalculated in light of the new m_{ij} determined in step 435.

[0071] In step 450, the difference between J_q^p and its predecessor value J_q^{p-1} is determined. If the difference is sufficiently small, as indicated in the inequality $|J_q^1 - J_q^0| < \varepsilon$, where ε is a small positive constant (here $\varepsilon = 0.0009$), then the process concludes at step 455, and the most recently determined $\{\mathbf{c}_j\}$ is used as the set of reference points for partitioning purposes. Otherwise, another set of centroids is determined in step 430. The process will continue until the condition of step 450 is satisfied.

[0072] Once the convergence criterion in step 450 has been met, the new centroids computed by Eq. 17 are used to partition the input data set into a set

of Voronoi cells. Such cells are illustrated in FIG. 5. A set 500 is shown partitioned into Voronoi cells, such as cells 505A through 505C. The Voronoi cells include centroids 510A through 510C respectively.

[0073] Once all the local networks are trained, additional patterns from the input set of patterns can be mapped into R^m as illustrated in FIG.6. The process begins with step 605. In step 610, the distance of the input pattern $\mathbf{x}$ to each reference point in $\{\mathbf{c}_i, i = 1,2,...c; \mathbf{c}_i \in R^n\}$ is determined. In step 615, the point $\mathbf{c}_j$ that is nearest to the input pattern $\mathbf{x}$ is identified. In step 620, the pattern $\mathbf{x}$ is mapped to a point $\mathbf{y}$ in R^m, $\mathbf{x} \rightarrow \mathbf{y}$, $\mathbf{x} \in R^n$, $\mathbf{y} \in R^m$ using the local neural network Net_j^L associated with the reference point $\mathbf{c}_j$ identified in step 615. The process concludes with step 625.

[0074] Note that new patterns in R^n that not in the original input set can also be projected into R^m in the manner shown in FIG. 6. Once the system is trained, new paterns in R^n are mapped by identifying the nearest local network and using that network in a feed-forward manner to perform the projection. An embodiment of a system that does this is illustrated in FIG. 7. The input for the system is a pattern 705 in R^n. This point is defined by its n attributes, (x_1, x_2, ...x_n). The system includes a dispatcher module 710, which compares the distance of the input point to the network centers (i.e., the reference points), and forwards the input point to one of the available local neural networks 701, 702, or 703. Specifically, the input pattern is sent to the local neural network associated with the reference point nearest to the input pattern. The chosen network then performs the final projection, resulting in an output point in R^m.

C. Local Nonlinear Mapping Networks – Algorithm III

[0075] The ability of a single network to reproduce the general structure of the nonlinear map suggests an alternative embodiment to overcome some of the complexities of clustering in higher dimensions. Conceptually, the alternative embodiment differs from Algorithm II in the way it partitions the data space. In contrast to the previous method, this process partitions the output space, and

clusters the training patterns based on their proximity on the *m*-dimensional nonlinear map rather than their proximity in the *n*-dimensional input space. For the training set, the assignment to a partition is straightforward. The images of the points in the training set on the nonlinear map are derived directly from the iterative algorithm described in Section II. For new points that are not part of the training set, the assignment is based on approximate positions derived from a global neural network trained with the entire training set, like the one described in section II. A. The general flow of the algorithm is similar to the one described in section II. B.

[0076] The training phase for this embodiment is illustrated in FIG. 8. The method begins at step 805. In step 810, a random set of patterns $\{\mathbf{x}_i, i = 1,2,\ldots,k; x_i \in R^n\}$ is extracted from the input data set. In step 815, the patterns $\mathbf{x}_i$ are mapped from R^n to R^m using the iterative nonlinear mapping algorithm described in section I B ($\mathbf{x}_i \rightarrow \mathbf{y}_i$, $i = 1,2,\ldots,k$, $\mathbf{x}_i \in R^n$, $\mathbf{y}_i \in R^m$). This mapping serves to define a training set T of ordered pairs $(\mathbf{x}_i, \mathbf{y}_i)$, $T = \{(\mathbf{x}_i, \mathbf{y}_i), i = 1,2,\ldots,k\}$.

[0077] In step 820, the points $\{\mathbf{y}_i, i = 1,2,\ldots,k, \mathbf{y}_i \in R^m\}$ are clustered into c clusters associated with c points in R^m, $\{\mathbf{c}_i, i = 1,2,\ldots c; \mathbf{c}_i \in R^m\}$. In the illustrated embodiment, fuzzy clusters are formed in this step using the FCM algorithm of FIG. 4. In step 825, the training set T is partitioned into c disjoint clusters C_j based on the distance of the images $\mathbf{y}_i$ from the cluster prototypes, $\{C_j = \{(\mathbf{x}_i, \mathbf{y}_i): d(\mathbf{y}_i, \mathbf{c}_j) \leq d(\mathbf{y}_i, \mathbf{c}_k)$ for all $k \neq j$; $j = 1,2,\ldots,c$; $i = 1,2,\ldots,k\}\}$. In step 830, c independent local neural networks $\{Net_i^L, i = 1,2,\ldots,c\}$ are trained with the respective clusters C_j derived in step 825. In step 835, a global network Net^G is trained with the entire training set T. The process concludes with step 840.

[0078] Once all the networks are trained, remaining input patterns from the input data set and any new patterns in R^n are projected using a tandem approach. An embodiment of this is illustrated in FIG. 9. The projection process begins at step 905. In step 910, each input pattern $\mathbf{x}$ to be projected

into R^m is mapped, $\mathbf{x} \rightarrow \mathbf{y}'$, $\mathbf{x} \in R^n$, $\mathbf{y}' \in R^m$, using the global network Net^G derived in step 835.

[0079] In step 915, the distance from $\mathbf{y}'$ to each reference point $\mathbf{c}_i$ in $\{\mathbf{c}_i, i = 1,2,...c; c_i \in R^m\}$ is determined. In step 920, the point $\mathbf{c}_j$ closest to $\mathbf{y}'$ is determined. In step 925, $\mathbf{x}$ is mapped into R^m, $\mathbf{x} \rightarrow \mathbf{y}$, using the local neural network associated with $\mathbf{c}_j$, Net_j^L. The process ends with step 930.

[0080] A system for performing the overall mapping $\mathbf{x} \rightarrow \mathbf{y}$ is shown in FIG. 10. First, an input pattern 1005 $\in R^n$ is projected by the global network 1010, Net^G, to obtain point 1012 $(\mathbf{y}') \in R^m$. Point $\mathbf{y}'$ can be viewed as having approximate coordinates on the nonlinear map. These coordinates are used to identify the nearest local network 1021 (Net_j^L) from among the possible local neural networks 1021 through 1023, based on the proximity of $\mathbf{y}'$ to each $\mathbf{c}_i$. Input point 1005 is projected once again, this time by the nearest local network 1021 to produce the final image 1030 on the display map.

III. Environment

[0081] The present invention may be implemented using hardware, software or a combination thereof and may be implemented in a computer system or other processing system. An example of such a computer system 1100 is shown in FIG. 11. The computer system 1000 includes one or more processors, such as processor 1104. The processor 1104 is connected to a communication infrastructure 1106 (e.g., a bus or network). Various software embodiments can be described in terms of this exemplary computer system. After reading this description, it will become apparent to a person skilled in the relevant art how to implement the invention using other computer systems and/or computer architectures.

[0082] Computer system 1100 also includes a main memory 1108, preferably random access memory (RAM), and may also include a secondary memory 1110. The secondary memory 1110 may include, for example, a hard disk drive 1112 and/or a removable storage drive 1114, representing a floppy disk

drive, a magnetic tape drive, an optical disk drive, etc. The removable storage drive 1114 reads from and/or writes to a removable storage unit 1118 in a well known manner. Removable storage unit 1118 represents a floppy disk, magnetic tape, optical disk, etc. As will be appreciated, the removable storage unit 1118 includes a computer usable storage medium having stored therein computer software and/or data. In an embodiment of the invention, removable storage unit 1118 can contain input data to be projected.

[0083] Secondary memory 1110 can also include other similar means for allowing computer programs or input data to be loaded into computer system 1100. Such means may include, for example, a removable storage unit 1122 and an interface 1120. Examples of such may include a program cartridge and cartridge interface (such as that found in video game devices), a removable memory chip (such as an EPROM, or PROM) and associated socket, and other removable storage units 1122 and interfaces 1120 which allow software and data to be transferred from the removable storage unit 1122 to computer system 1100.

[0084] Computer system 1100 may also include a communications interface 1124. Communications interface 1124 allows software and data to be transferred between computer system 1100 and external devices. Examples of communications interface 1124 may include a modem, a network interface (such as an Ethernet card), a communications port, a PCMCIA slot and card, etc. Software and data transferred via communications interface 1124 are in the form of signals 1128 which may be electronic, electromagnetic, optical or other signals capable of being received by communications interface 1124. These signals 1128 are provided to communications interface 1124 via a communications path (i.e., channel) 1126. This channel 1126 carries signals 1128 and may be implemented using wire or cable, fiber optics, a phone line, a cellular phone link, an RF link and other communications channels. In an embodiment of the invention, signals 1128 can include input data to be projected.

[0085] In this document, the terms "computer program medium" and "computer usable medium" are used to generally refer to media such as removable storage drive 1114, a hard disk installed in hard disk drive 1112, and signals 1128. These computer program products are means for providing software to computer system 1100. The invention is directed to such computer program products.

[0086] Computer programs (also called computer control logic) are stored in main memory 1108 and/or secondary memory 1110. Computer programs may also be received via communications interface 1124. Such computer programs, when executed, enable the computer system 1100 to perform the features of the present invention as discussed herein. In particular, the computer programs, when executed, enable the processor 1104 to perform the features of the present invention. Accordingly, such computer programs represent controllers of the computer system 1100.

[0087] In an embodiment where the invention is implemented using software, the software for performing the training and projection phases of the invention may be stored in a computer program product and loaded into computer system 1100 using removable storage drive 1114, hard drive 1112 or communications interface 1124.

[0088] In another embodiment, the invention is implemented using a combination of both hardware and software.

VI. Conclusion

[0100] While various embodiments of the present invention have been described above, it should be understood that they have been presented by way of example, and not limitation. It will be apparent to persons skilled in the relevant art that various changes in detail can be made therein without departing from the spirit and scope of the invention. Thus the present invention should not be limited by any of the above described exemplary

embodiments, but should be defined only in accordance with the following claims and their equivalents.